Sudarshan Angirash

Enterprise & Solution Architect | GenAI, Cloud, Multi-Agent Systems | 20 Years Building Reliable, High-Impact Platforms
sangirash@gmail.com | +91 9354420046 | Hyderabad, Telangana, India
https://www.linkedin.com/in/sangirash/ || https://github.com/sangirash

Why Me?

Over nearly two decades in technology architecture and enterprise transformation, I've learned that the most elegant solutions rarely look elegant on the surface. They're messy, iterative, and grounded in what actually matters: does it work, and does it move the business forward? I design and govern enterprise architectures across large-scale digital transformation programs in complex, cloud-native, AI-first, heavily regulated environments. My expertise spans multi-cloud infrastructure, integration patterns, GenAI, and multi-agent systems but the real skill is translating architectural complexity into measurable business outcomes: reliability, velocity, cost, and user experience. That means navigating senior stakeholder dynamics, communicating vision to non-technical leadership, and establishing governance frameworks that stick rather than become expensive paperwork. I've spent countless hours explaining to C-suite executives why their preferred architecture creates technical debt, then finding compromises that preserve system integrity while securing buy-in.

I've led platforms handling 15,000+ daily customer interactions with 95%+ automated resolution rates. Miss that target slightly and support costs explode; exceed it and you've transformed customer experience. I've also cut delivery timelines by 30–40% through specific architectural decisions within 2 months: enabling parallelization, shortening feedback loops, eliminating deployment bottlenecks. Infrastructure tells the same story. Architecting MLOps, DevOps, and SRE capabilities for mission-critical environments improved system uptime and reduced operational costs by over 50% by addressing both technical tooling (infrastructure-as-code, CI/CD automation, monitoring frameworks) and non-technical barriers (process, team structure, information flow).

For AI systems, in 11 months I architect and implement production multi-agent deployments using LangGraph for workflow composition, PyTorch for model fine-tuning and inference optimization, and LLM providers (OpenAI, Anthropic, open-source) selected on latency, cost, and capability trade-offs. I've built reliable AI layers handling 15,000+ interactions where 95%+ automation requires more than clever prompts: token economics optimization, retrieval-augmented generation (RAG) with vector embeddings (Pinecone, Weaviate, self-hosted), confidence thresholding, and fallback strategies. Prompt engineering becomes infrastructure through version control, A/B testing, and structured JSON outputs.

Integration is where most AI projects fail. I design API-first architectures with containerized AI services (Kubernetes) connected to operational systems via event-driven patterns or synchronous microservices (Python, Go, Node.js). Critical decisions: does your RAG need hourly refresh or nightly batch? How do you version embeddings when models change? I've navigated these across regulated environments (FedRAMP High, HIPAA-adjacent), multi-tenant SaaS, and internal platforms. Governance, audit trails, user explainability, prediction drift detection is architected in from the start, not bolted on. MLOps tooling (model promotion CI/CD, drift monitoring) keeps this sustainable rather than a manual burden.

The core work remains constant: translating between worlds. Between what's technically possible and commercially viable. Between innovation and stability. Between theory and what works when production is down at 2am. That translation skill more than any specific technology is what 20 years has taught me.

PROFESSIONAL SUMMARY

My work centres on defining what enterprise and solution architectures should look like, then translating that into roadmaps people can actually execute. I build target state definitions, architecture decision records, transition plans. But the documentation only matters if it moves the needle on what the business cares about—deployment frequency, system reliability, cost, customer experience. On AWS, Kubernetes, with Terraform, Python, and whatever else gets the job done.

I led a multi-agent customer support platform that processes 15,000 interactions daily. It resolves roughly 95% of those automatically, slashing manual support overhead by 40–50% and cutting time-to-resolution by 30%. That system runs on AWS EKS, uses LangGraph and PyTorch for the AI layer, connects via microservices and event-driven patterns. It's stable because
we invested in the unglamorous work: SRE blueprints, CI/CD automation with infrastructure-as-code, metrics-driven release processes. We ship bi-weekly. Rollbacks stay below 1%.

The MLOps and DevOps transformations I've led follow a similar thread. Most teams inherit "monthly releases if we're lucky" or quarterly. Moving that to bi-weekly requires rethinking infrastructure (EKS, serverless ETL), automation (proper CI/CD), and culture. I've cut operational incidents by 20–30% in those transitions. System availability? We hit 99.9%+. Not through heroic debugging at 3am, but through architecture that makes reliability boring and systematic.

I've worked across domains: HR tech, cloud services, security/compliance (FedRAMP High work), data integration, AI/Data platforms, gaming, fintech, and e-commerce. That breadth matters because architecture isn't domain-specific, it's about understanding trade-offs. When should you split a monolith? When is that "microservices architecture" actually making your life harder? When does that shiny new framework solve a real problem versus creating maintenance headaches?

The part of this work I've come to value most is the collaboration piece. I spend a lot of time in architecture workshops with product, engineering, operations, and security teams simultaneously. Aligning people across functions is harder than solving the technical problems. I document the options, show the risks, present the trade-offs clearly. Sometimes the best architecture loses to the one the CEO's already

decided on, learning when that matters and when to fight for it is experience you only get by doing it for 19 years. But my decisions and approach always had been data driven.

I also mentor other architects and engineering leaders to think systematically about these decisions rather than defaulting to "everyone's using it" or "it's in the task description" Good architecture practice spreads through the team when the person modelling it can explain the reasoning. That's part of the job too.

CORE ARCHITECTURE VALUE PROPOSITION

Enterprise & Solution Architecture Leadership
I define target and transition architectures, not just diagrams, but actual, executable roadmaps with milestones and dependencies. Recent work includes an AI-powered customer support platform (application, data, integration, infrastructure layers). I maintain architecture decision logs and reference architectures that give teams a shared language instead of everyone reinventing. One client reduced design divergence across teams by 30% just by having standards enforced and communicated consistently.

Large-Scale Digital Transformation & Cloud
Moving from quarterly releases to bi-weekly wasn't a weekend project. It required rearchitecting for containers (EKS, not EC2), adopting serverless ETL where it made sense, and automating every step of the pipeline. I've done this multiple times now. The first time, it took six months. The playbook gets faster, but you can't skip the hard parts. The payoff: deployment frequency increases an order of magnitude. Stringent security and compliance requirements? I've navigated FedRAMP High, handled multi-tenancy, worked through audit requirements. The architecture has to satisfy both.

AI/Data, Integration & Governance
End-to-end data and AI systems live in my wheelhouse. I've built LangGraph-based multi-agent systems, worked with PyTorch and SageMaker, embedded vector stores into existing applications. The tricky part is integration: existing systems talk via APIs, microservices (Python, Go, Node.js backends), or message queues. The architecture sits at that intersection. I've also facilitated the governance piece, ensuring decisions about data flow, security, and operational ownership are made before you're six months into the project and someone realizes nobody agreed on who manages the deployment pipeline.

PROFESSIONAL EXPERIENCE

Founder and CTO
TechTonic IT Solutions (https://techtonicis.com)
01/2025 - Present

- Built ATSandYou (https://atsandyou.com), an AI-powered resume optimizer delivered end-to-end within 9 days (MVP) plus LinkedIn feature in 2 more days (fully function production rollout was done within 2 months). The technical stack: React frontend, NLP embeddings and vector search on AWS, microservices architecture with infrastructure-as-code and automated CI/CD pipelines. The architecture decision was critical early: how do you version embeddings? How do you A/B test matching algorithms without slowing down iteration? We solved that through rapid experimentation frameworks. The outcome: JD-resume match accuracy improved 25–30% while cutting deployment effort by 80%. Led the full product lifecycle architecture, development, QA, analytics, SEO and reduced defects by 20% by establishing observability and security standards that stuck.

- Architected and delivered YudisAI (https://yudis.techtonicis.com), a multi-tenant SaaS platform for enterprise performance management and HR transformation within 5 months. The scope was ambitious: real-time performance data pipelines, 20+ third-party integrations (GitHub, Jira, Salesforce, Slack, OpenAI, Claude), custom LLM support with on-premises options, row-level security for multi-tenancy. The platform generates AI-driven employee profiles, predictive attrition forecasting, bell curve dashboards, and configurable rating cycles; all with bias-free evaluation logic and employee self-service capabilities. I positioned it at 1/10th the cost of legacy HR systems while maintaining enterprise-grade security and compliance. CXOs could actually plan workforce changes with transparent, data-driven insights instead of guessing. That required careful architecture: the data layer handles massive-scale aggregations, the AI layer processes unstructured performance data, and the real-time pipeline ensures dashboards don't lag behind reality by days.

- As the CTO and lone force behind product development, growth, and day-to-day operations, I single-handedly took ATSandYou from nothing to $3,200 in monthly recurring revenue, with 1,200 active users, all in just four months. I handled every part of the sales pipeline myself, ran targeted marketing pushes, dialled in SEO that actually delivered, and rigged up automated outreach sequences in Apollo.io to supercharge D2C leads. For both ATSandYou (D2C) and Yudis (B2B), I crafted and fine-tuned the full sales funnel: closing deals directly, smoothing out onboarding, and iterating on the product nonstop. Blending deep technical know-how with straight-up revenue hustling let me build real momentum and steady income growth.

AI Agent creator, Author, MCP researcher
Freelancer
11/2023 - 12/2024

- Designed and implemented multi-agent GenAI architectures for production clients. The pattern I've built relies on combining LLMs (OpenAI, Anthropic, open-source) with Rust-based backends and tool-calling via RMCP for system integration. This matters because the architecture determines whether your AI layer becomes a bottleneck or an asset. I've handled retrieval-augmented generation

(RAG) at scale, external system orchestration through event-driven patterns, and the orchestration complexity that comes when a single agent needs to coordinate across five different APIs. The throughput and reliability improvements come from architectural choices: how you batch requests, where you cache embeddings, whether you process in parallel or sequentially based on dependencies.

- Created reusable architecture patterns and reference implementations for integrating large language models with legacy enterprise data sources and APIs. Most organizations have years of data trapped in systems never designed for AI; Salesforce, ERP systems, data warehouses. The challenge isn't the LLM; it's building connectors that don't choke under load and don't require rearchitecting your entire data layer. I've worked through those trade-offs: real-time API calls versus nightly ETL batch updates, caching strategies for frequently accessed contexts, security models that prevent prompt injection attacks. This work reduced integration rework effort by an estimated 25% for clients across healthcare, fintech, and SaaS verticals.

- Built **ProMailHunter** and **Saangi** as full-stack developer, production-grade platforms, built from solo architecture design through deployment. ProMailHunter: (https://promailhunter.com) system design for email address discovery, data modelling for deduplication and verification, monitoring and alerting so you know when accuracy drifts, developer and went live within 2 week. Saangi: (https://saangi.com) similar scope but for e-commerce domain. From concept to production within 45 days. In Saangi most of the automation was done using n8n. That velocity only happens when you've made the architectural choices that matter and can execute without debate. Continuous monitoring, rapid iteration, automated testing. You can't afford to debug in production when you're shipping solo.

- **Authored two technical books**: one on multi-agent AI strategies (https://amzn.in/d/7hgzUi2) and one on advanced Rust for systems programming (https://a.co/d/4Xsc7rE). The first codifies architectural patterns: how do you structure agents that don't hallucinate? How do you handle tool-calling failures? How do you make the system explainable? The second focuses on building reliable, low-latency backends for data-intensive workloads.

- Advised engineering teams directly on architecture choices: cloud infrastructure, data pipeline design, AI integration strategy. Conducted focused architecture reviews not rubber-stamp approval, but working through the actual trade-offs. When should you use Kubernetes versus managed services? When does that microservices split actually reduce complexity versus creating a distributed debugging nightmare? How do you reason about observability in an AI-heavy system? Led hands-on workshops to uplift teams' ability to reason through these decisions themselves rather than depend on outside architects. The goal: teams that can make good architectural choices even when external advisors aren't in the room.

System Development Manager III

Amazon Web Services, Amazon
05/2022 - 11/2023

- Led architecture and delivery for a multi-agent customer support system with in 11 months. We processed 15k+ daily queries. Used PyTorch to train models. LangGraph handled agent coordination. AWS ran the infrastructure. Hit 95% accuracy on classification and resolution. Manual support dropped 40-50%. Customers got faster responses. The hard part? Making sure the system knew when to escalate to humans instead of guessing wrong.

- Mapped out the migration for ML workflows to EKS. Defined what the end state looks like. Container strategy. Autoscaling rules that don't flip-flop. Observability so you see problems early. Infrastructure costs got steadier. Estimated 20% less volatility. Scalability improved. No more guessing if the cluster will hold up.

- Brought in SageMaker. Set up MLOps properly. Versioned features and data. Added monitoring for drift. Rollback plans ready. Models went from weeks to days for deployment. Fewer incidents in production. That's what happens when you treat models like code. Also integrated Amazon Bedrock for serverless LLM inference and custom model fine-tuning. Picked it when we needed quick access to multiple foundation models without managing infrastructure. Bedrock handled the heavy lifting for agent tool-calling and response generation in high-volume scenarios. Combined it with EKS for hybrid workloads, Bedrock for bursty LLM calls, EKS for custom PyTorch models. Cost stayed predictable. No vendor lock-in headaches. Made switching between Claude and Llama3 seamless.

- Overhauled release management. Bi-weekly deploys now. Frequency doubled or tripled. Rollbacks under 1%. Pipelines rethought from scratch. Metrics decide what ships. Governance didn't slow things down. Business goals actually made it into releases. Built modules with GraphDB clients. Reinforcement learning for decisions. Go microservices for speed. Autoscaling based on real health checks. Utilization got better. Intelligent routing across clouds.

- Worked side-by-side with product leads, operations, security to reviewed decisions together, discuss trade-offs, and approvals. Architecture stayed practical. Matched business needs and rules. No last day surprises.

Prior Experience

- **Oct'17 – May'22,** UKG (Kronos) as Sr DevOps Manager
- **Nov'15 – Oct'17,** OSI Consulting as Release Manager
- **Dec'12 – Nov'15,** Cognizant Technology Solutions as Manager Operations
- **Aug'12 – Dec'12,** Varite India Pvt. Ltd., Technical Lead (Build and Release)
- **Jan'11 – Aug'12,** Tribal Fusion (Exponential), Technical Lead
- **Jun'06 – Jul'10,** Sapient Corporation Pvt. Ltd., Sr. Associate Technology L1

SKILLS

Enterprise & Solution Architecture:
Target and transition state definition, architecture roadmaps, architecture decision records (ADRs), architecture governance and review forums (ARB, EARB, CAB), reference architectures, patterns and standards.

Cloud & Platforms:
AWS (EKS, SageMaker, CloudWatch, Bedrock, Glue, serverless), Kubernetes, Docker, Terraform, CI/CD (Jenkins, GitHub Actions), hybrid cloud and on-prem integration, multi-region/high-availability designs.

AI/Data & Multi-Agent Architectures:
Generative AI, Agentic AI, LangGraph, PyTorch, TensorFlow, scikit-learn, embeddings, RAG, NLP, data pipelines, feature stores, model deployment and monitoring, multi-agent orchestration patterns.

Integration & Application Architecture:
Microservices (Go, Node.js, Python), REST APIs, event-driven patterns (Kafka), GraphDB integration, API gateways and security, asynchronous workflows, fault-tolerant design.

Full stack developer:
Frontend (React, Vue.js, Next.js, Typescript), Backend (Node.js, Spring Boot, Python, Go, Django), Database (PostgreSQL, MongoDB, GraphDB, SQLite).

Day to Day AI tool uses:
Research (Perplexity), Image/Video (Heygen, Gemini, Canva), Automation (n8n), Coding (Claude code), Reasoning/Planning (ChatGPT, Claude), Sales and Leads Pipeline (Apollo.io)

DevOps, SRE & Operations:
MLOps, DevOps automation, site reliability engineering, observability (Grafana, Splunk), incident management and SLIs/SLOs, capacity planning, release management.

Security, Compliance & Governance:
FedRAMP High aligned architectures (Aurora-based solutions), security and vulnerability response workflows, compliance management, audit-ready logging and traceability.

Leadership & Stakeholder Management:
Leading architecture workstreams and cross-functional squads, stakeholder alignment, C-level and senior leadership communication, facilitation of architecture workshops, mentoring engineers and managers.

EDUCATION

MSc Information Technology
University of Rajasthan, Jaipur, India
2004
BSc Mathematics
Mohanlal Sukhadiya University, Udaipur, India
2001

CERTIFICATIONS

ITIL 2011 Foundation Certified - ITIL
MongoDB Certified - MongoDB